Java Chapter 5: Methods

Contents

Time required: 90 minutes

DRY

Don't **R**epeat **Y**ourself (**DRY**) is a principle of software engineering aimed at reducing repetition of software patterns. It you are repeating any code, there is probably a better solution.

Read: Think Java

- Chapter 4 Methods and Testing

Online Tutorials

- Java Methods
- Java Method Parameters
- Java Method Overloading
- Java Scope
- Java Recursion
- Java Math Methods

- Java Random Numbers

Tutorial 5.1 – Void Method

Time to create our first method. This program has two methods, `message()` and `main()`. The `main()` method is where the program starts.

Put in your own favorite saying in this tutorial.

```
1  // Name: VoidMessageMethod.java
2  // Written by:
3  // Written on:
4  // Purpose: Void method
5
6  public class VoidMessageMethod {
7     public static void main(String[] args) {
8        System.out.println("First method call");
9        // Method call
10       message();
11       System.out.println("\nSecond method call");
12       // Method call
13       message();
14       System.out.println("\nThird method call");
15       // Method call
16       message();
17    }
18    // Create void method
19    static void message() {
20       System.out.println("It is not necessary to change. Survival is not mandatory.");
21       System.out.println("W. Edwards Deming");
22    }
23 }
24
25
26
```

Example run:

```
First method call
It is not necessary to change. Survival is not mandatory.
W. Edwards Deming

Second method call
It is not necessary to change. Survival is not mandatory.
W. Edwards Deming

Third method call
It is not necessary to change. Survival is not mandatory.
W. Edwards Deming
```

Tutorial 5.2 – Method with a Single Argument

The following program has a method with a single argument

Create a Java program named: **MethodSingleArgument.java**

```java
/**
 * Name: MethodSingleArgument.java
 * Written by:
 * Written on:
 * Purpose: Method with 1 double argument
 */

// Import Math library for sqrt method
import java.lang.Math;

public class MethodSingleArgument {
   public static void main(String[] args) {
      double num1 = 4;
      double num2 = 8.9;
      double num3 = 112;

      // Method call with 1 double parameter
      findSquareRoot(num1);
      findSquareRoot(num2);
      findSquareRoot(num3);
   }

   // void method with 1 double argument
   static void findSquareRoot(double num) {
      double squareRoot;
      squareRoot = Math.sqrt(num);
      System.out.println("The square root of " + num + " is: " + squareRoot);
   }
}
```

Example run:

```
The square root of 4.0 is: 2.0
The square root of 8.9 is: 2.9832867780352594
The square root of 112.0 is: 10.583005244258363
```

Tutorial 5.3 – Method with Multiple Arguments

The following program demonstrates a function with 2 datatypes and 3 arguments.

Create a Java program named: **MethodMultipleArguments.java**

```java
/**
 * Name: MethodMultipleArguments.java
 * Written by:
 * Written on:
 * Purpose: Method with 2 double and one String argument
 */

public class MethodMultipleArguments {
   public static void main(String[] args) {
      String message = "\nFirst method call";

      // Method call with 2 double and 1 String parameter
      multiply(2, 3.2, message);

      message = "\nSecond method call";
      // Method call with 2 double and 1 String parameter
      multiply(4, 313, message);

      message = "\nThird method call";
      // Method call with 2 double and 1 String parameter
      multiply(20.1, 23.521, message);
   }

   // void method with 2 double arguments
   static void multiply(double num1, double num2, String message) {
      double product;
      product = num1 * num2;
      System.out.println(message);
      System.out.println("The product of " + num1 + " * " + num2 + " is: " + product);
   }
}
```

Example run:

```
First method call
The product of 2.0 * 3.2 is: 6.4

Second method call
The product of 4.0 * 313.0 is: 1252.0

Third method call
The product of 20.1 * 23.521 is: 472.7721
```

Tutorial 5.4 – Method with Return Value

Methods can do some work and return a value.

Create a Java program named: **MethodReturnValue.java**

```java
/**
 * Name: MethodReturnValue.java
 * Written by:
 * Written on:
 * Purpose: Method with a return value
 */

// Import Math library for ceil method
import java.lang.Math;

public class MethodReturnValue {
   public static void main(String[] args) {
      double num1 = 4.54;
      double num2 = 8.2;
      double ceil2;

      // Method call with parameter and return value
      System.out.println("\nPrint method value return directly");
      System.out.println(getCeil(num1));

      System.out.println("Assign value return to variable");
      ceil2 = getCeil(num2);
      System.out.println(ceil2);
   }

   // method with 1 double argument
   // and 1 double return value
   static double getCeil(double num) {
      double numCeil;
      // ceil rounds a floating point
      // up to the nearest integer value
      numCeil = Math.ceil(num);
      return numCeil;
   }
}
```

Example run:

```
Print method value return directly
5.0
Assign value return to variable
9.0
```

Assignment Submission

1. Attach the pseudocode.
2. Attach the program files.

3. Attach screenshots showing the successful operation of the program.
4. Submit in Blackboard.